

Extensible Records in the System E Framework and a New Approach to Object-Oriented Type Inference

Ryan Heise

A dissertation submitted for the degree of Doctor of Philosophy

Faculty of Information Technology

University of Technology, Sydney

2013

Copyright © 2013

Ryan Heise

Abstract

Extensible records were proposed by Wand as a foundation for studying object-oriented type inference. One of their key benefits is that they allow for an elegant encoding of object-oriented inheritance, where one class of objects may be defined as an extension of another class of objects. However, every system of type inference designed for extensible records to date has been developed in the Hindley/Milner-style, a consequence being that polymorphism in these systems is not first-class and analysis is not strictly compositional. We argue that both of these features are necessary to retain the modelling and engineering benefits of traditional object-oriented languages such as Java:

1. Object-oriented modelling depends on the treatment of objects as first-class citizens, and this demands a type inference system capable of handling first-class polymorphism.
2. Object-oriented engineering encourages the separate development of software modules, and this should be supported by the type inference system with compositional analysis.

Both of these features are present in a type system for the λ -calculus called System E, which supports first-class polymorphism via intersection types, and compositional type inference via expansion variables. However, research into System E has so far focused on refining and simplifying the formulation of expansion variables and exploring type inference algorithms with various properties. Meanwhile, the system has not yet been extended beyond the terms of the pure λ -calculus and it lacks many features that would be needed in a practical object-oriented language.

In this dissertation, we combine System E with Wand's extensible records resulting in a new approach to type inference for extensible records that better preserves the modelling and engineering benefits of object orientation stated above. The resulting system, called System E^{vcr}, is significant because previous

type inference systems, both for extensible records in particular, and also for object orientation in general, have at best preserved only one or the other of these two benefits, but never both of them simultaneously. System E^{vcr} also makes a significant contribution to the work on System E, since it demonstrates for the first time that the System E's expansion variables can be adapted to analyse programs whose term language extends beyond the pure λ -calculus.

To demonstrate the potential use of System E^{vcr} in object-oriented type inference, an implementation of our type inference algorithm was created and is shown to succeed on problem examples that previous systems either fail to analyse, or else fail to analyse compositionally.

Declaration

I declare that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text. I also declare that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I declare that all information sources and literature used are indicated in the thesis.

(Ryan Heise)

Acknowledgements

I wish to thank my principal supervisor Barry Jay and co-supervisor Wayne Brooks for their guidance and support, and for teaching me the toolset that I depended upon to develop the ideas within this dissertation. I also wish to thank Pascal Zimmer for helping me to understand the capabilities of existing extensible record type systems, Joe Wells for many vital and informative discussions about expansion variables and Adam Bakewell for assisting me to understand the **opus** unification algorithm. Many thanks also go to fellow student Quy Tuan Nguyen with whom I have had many stimulating discussions about both his own research and mine.

This work was supported by an Australian Postgraduate Award from the Australian Government Department of Innovation, Industry, Science and Research.

Table of Contents

1	Introduction	1
1.1	Key Considerations for Object-Oriented Type Inference	4
1.1.1	Object-Orientation	5
1.1.2	First-Class Polymorphism	8
1.1.3	Compositionality	18
1.2	Prior Work	23
1.2.1	Object-Orientation & First-Class Polymorphism	23
1.2.2	Object-Orientation & Compositionality	24
1.2.3	First-Class Polymorphism & Compositionality	26
1.3	Extensible Records in the System E Framework	30
1.3.1	Extensible Records	30
1.3.2	System E	34
1.3.3	System E ^{vcr}	35
1.4	Outline of the Dissertation	39
1.5	How to read this dissertation	40
2	Review of System E	43
2.1	λ -Calculus	44
2.2	Type System	45
2.3	Expansions and Expansion Variables	49
2.3.1	The Traditional Approach	50
2.3.2	The System E Approach	52
2.4	Type Inference	56
2.4.1	β -Unification	57
2.4.2	Covering Unification	64
2.4.3	Comparison of β -Unification and Covering Unification	67
2.5	Summary	68

3	System E^v: The Value Restriction	70
3.1	The Value Restriction	71
3.2	Simplifications	72
3.2.1	Type Equivalences	72
3.2.2	Restrictions to Simple Types	73
3.2.3	Unique Derivations	74
3.3	Type System	75
3.3.1	Terms and Reductions	75
3.3.2	Types and Expansions	77
3.3.3	Typing Derivations	85
3.4	Type Inference	99
3.4.1	Preliminary Definitions	100
3.4.2	Algorithm $\text{opus}\beta$	101
3.4.3	Algorithm \mathcal{I}	110
3.5	Examples	122
3.6	Efficiency	125
3.7	Summary	126
4	System E^{vc}: Constants	128
4.1	Motivations	128
4.2	Integrating Constants with E-Variables	129
4.3	Type System	130
4.3.1	Terms and Reductions	130
4.3.2	Types and Expansions	132
4.3.3	Typing Derivations	134
4.4	Type Inference	140
4.4.1	Algorithm $\text{opus}\beta$	140
4.4.2	Algorithm \mathcal{I}	141
4.5	Examples	143
4.5.1	Issues	146
4.6	Summary	147
5	System E^{ver}: Extensible Records	148
5.1	Integrating Extensible Records with E-Variables	148
5.2	Type System	153
5.2.1	Terms and Reductions	153
5.2.2	Types and Expansions	156
5.2.3	Typing Derivations	163
5.3	Type Inference	171

TABLE OF CONTENTS

vii

5.3.1	Algorithm $\text{opus}\beta$	171
5.3.2	Algorithm \mathcal{I}	176
5.4	Examples	183
5.4.1	Extensible Records	183
5.4.2	Object-Orientation	187
5.4.3	First-Class Polymorphism	194
5.4.4	Compositionality	197
5.5	Efficiency	200
5.6	Summary	201
6	Conclusions	203
6.1	Future Work	205
	Bibliography	207
A	Additional Proofs	214
B	Examples from the System E Inference Report	228
C	The opus Unification Algorithm	234
C.1	Adapting opus to System E^{vcr}	236

List of System E^\vee Definitions and Theorems

Definition 3.1 (Terms for E^\vee)	75
Definition 3.2 (Free variables for E^\vee)	76
Definition 3.3 (Term substitution for E^\vee)	76
Definition 3.4 (α -conversion for E^\vee)	77
Definition 3.5 (Reduction for E^\vee)	77
Definition 3.6 (Syntax for E^\vee)	77
Definition 3.7 (Type equivalence for E^\vee)	78
Definition 3.8 (Intersection components for E^\vee)	79
Lemma 3.9 (Intersection components for E^\vee)	79
Corollary 3.10 (Simple type equivalence for E^\vee)	80
Definition 3.11 (Expansion application for E^\vee)	80
Lemma 3.12 (Expansion application preserves syntactic categories for E^\vee)	81
Lemma 3.13 (\Box acts as the identity for E^\vee)	81
Lemma 3.14 (Expansion composition for E^\vee)	81
Lemma 3.15 (Expansion distribution equivalence for E^\vee)	82
Lemma 3.16 (Expansion preserves type equivalence for E^\vee)	83
Lemma 3.17 (Type variables for E^\vee)	85
Corollary 3.18 (Type substitution for E^\vee)	85
Definition 3.19 (Term contexts for E^\vee)	85
Definition 3.20 (Operations on term contexts for E^\vee)	86
Definition 3.21 (Typings for E^\vee)	86
Definition 3.22 (Operations on typings for E^\vee)	86
Definition 3.23 (Typing judgements for E^\vee)	87
Definition 3.24 (Typing derivations for E^\vee)	87
Lemma 3.25 (Typing derivations for E^\vee)	88

Lemma 3.26 (Variable types for E^V)	88
Lemma 3.27 (Equivalent types for E^V)	89
Lemma 3.28 (Expansion for E^V)	91
Lemma 3.29 (Term substitution for E^V)	94
Theorem 3.30 (Subject reduction for E^V)	97
Theorem 3.31 (Progress for E^V)	98
Definition 3.32 (Distinct sequences for E^V)	100
Definition 3.33 (Unification constraints for E^V)	100
Definition 3.34 (Operations on unification constraints for E^V)	100
Definition 3.35 (Solved unification constraints and constraint sets for E^V)	101
Definition 3.36 (Unifiers for E^V)	101
Definition 3.37 (Unification algorithm for E^V)	101
Definition 3.38 (Covering unifier sets for E^V)	101
Definition 3.39 (Covering unification algorithm for E^V)	101
Definition 3.40 (rfactor_β for E^V)	102
Lemma 3.41 (Correctness of $\overrightarrow{\text{rfactor}_\beta}$ for E^V)	102
Definition 3.42 (Variable structures for E^V)	104
Definition 3.43 (varstruct for E^V)	105
Definition 3.44 (ovars for E^V)	106
Definition 3.45 (Fresh renamings for E^V)	107
Definition 3.46 ($\overrightarrow{\text{opus}_\beta}$ relation for E^V)	107
Definition 3.47 (opus_β reduction for E^V)	109
Lemma 3.48 (Correctness of \Rightarrow_*^σ for E^V)	109
Definition 3.49 (Algorithm opus_β for E^V)	110
Theorem 3.50 (Correctness of opus_β for E^V)	110
Definition 3.51 (estrip for E^V)	110
Lemma 3.52 (Correctness of estrip for E^V)	111
Lemma 3.53 (estrip reversion for E^V)	111
Definition 3.54 (isect for E^V)	112
Lemma 3.55 (Correctness of isect for E^V)	112
Lemma 3.56 (isect reversion for E^V)	113
Definition 3.57 (Algorithm \mathcal{I} for E^V)	115
Theorem 3.58 (Termination of \mathcal{I} for E^V)	116
Theorem 3.59 (Correctness of \mathcal{I} for E^V)	116
Theorem 3.60 (Principality of \mathcal{I} for E^V)	118

List of System E^{vc} Definitions and Theorems

Amendment to Definition 3.1 (Terms for E^{vc})	130
Amendment to Definition 3.2 (Free variables for E^{vc})	131
Amendment to Definition 3.3 (Term substitution for E^{vc})	131
Amendment to Definition 3.5 (Reduction for E^{vc})	132
Amendment to Definition 3.6 (Syntax for E^{vc})	132
Amendment to Definition 3.11 (Expansion application for E^{vc})	133
Definition 4.1 (typeof for E^{vc})	134
Amendment to Definition 3.24 (Typing derivations for E^{vc})	134
Requirement 4.2 (ϕ -typability for E^{vc})	135
Lemma 4.3 (Constant types for E^{vc})	135
Lemma 4.4 (Constant term contexts for E^{vc})	136
Restatement of Lemma 3.25 (Typing derivations for E^{vc})	137
Restatement of Lemma 3.27 (Equivalent types for E^{vc})	137
Restatement of Lemma 3.28 (Expansion for E^{vc})	138
Restatement of Lemma 3.29 (Term substitution for E^{vc})	138
Restatement of Theorem 3.30 (Subject reduction for E^{vc})	139
Restatement of Theorem 3.31 (Progress for E^{vc})	139
Amendment to Definition 3.43 (varstruct for E^{vc})	140
Restatement of Theorem 3.50 (Correctness of opusβ for E^{vc})	141
Amendment to Definition 3.57 (Algorithm \mathcal{I} for E^{vc})	141
Restatement of Theorem 3.58 (Termination of \mathcal{I} for E^{vc})	142
Restatement of Theorem 3.59 (Correctness of \mathcal{I} for E^{vc})	142
Restatement of Theorem 3.60 (Principality of \mathcal{I} for E^{vc})	143

List of System E^{vcr} Definitions and Theorems

Amendment to Definition 3.1 (Terms for E^{vcr})	153
Amendment to Definition 3.2 (Free variables for E^{vcr})	154
Amendment to Definition 3.3 (Term substitution for E^{vcr})	154
Amendment to Definition 3.5 (Reduction for E^{vcr})	155
Amendment to Definition 3.6 (Syntax for E^{vcr})	156
Definition 5.1 (Meets judgement for E^{vcr})	157
Definition 5.2 (Valid substitutions for E^{vcr})	157
Convention 5.3 (Valid substitutions for E^{vcr})	157
Lemma 5.4 (Meets subsumption for E^{vcr})	157
Lemma 5.5 (Meets substitution for E^{vcr})	158
Definition 5.6 (Lacks judgement for E^{vcr})	159
Lemma 5.7 (Lacks equivalence for E^{vcr})	159
Amendment to Definition 3.11 (Expansion application for E^{vcr})	161
Lemma 5.8 (Lacks expansion for E^{vcr})	161
Amendment to Definition 4.1 (typeof for E^{vcr})	163
Amendment to Definition 3.24 (Typing derivations for E^{vcr})	164
Lemma 5.9 (Label identity for E^{vcr})	165
Lemma 5.10 (Extension parameter type for E^{vcr})	165
Restatement of Lemma 3.25 (Typing derivations for E^{vcr})	166
Restatement of Lemma 3.27 (Equivalent types for E^{vcr})	167
Restatement of Lemma 3.28 (Expansion for E^{vcr})	168
Restatement of Lemma 3.29 (Term substitution for E^{vcr})	169
Restatement of Theorem 3.30 (Subject reduction for E^{vcr})	169
Restatement of Theorem 3.31 (Progress for E^{vcr})	170
Amendment to Definition 3.43 (varstruct for E^{vcr})	172

Amendment to Definition 3.45 (Fresh renamings for E^{vcr})	172
Definition 5.11 (hunify for E^{vcr})	172
Lemma 5.12 (hunify correctness for E^{vcr})	173
Lemma 5.13 (hunify is complete and principal for E^{vcr})	173
Amendment to Definition 3.46 ($\overrightarrow{\text{opus}\beta}$ relation for E^{vcr})	175
Restatement of Theorem 3.50 (Correctness of $\text{opus}\beta$ for E^{vcr})	176
Amendment to Definition 3.57 (Algorithm \mathcal{I} for E^{vcr})	176
Lemma 5.14 (Row variable substitution for E^{vcr})	178
Restatement of Theorem 3.58 (Termination of \mathcal{I} for E^{vcr})	179
Restatement of Theorem 3.59 (Correctness of \mathcal{I} for E^{vcr})	179
Restatement of Theorem 3.60 (Principality of \mathcal{I} for E^{vcr})	180

List of Figures

2.1	System E type equalities	47
2.2	System E typing rules	48
2.3	System E expansion application rules	54
3.1	Equivalences considered by the implementation of $\mathbf{rfactor}_\beta$	105
3.2	Performance of \mathbf{opus}_β vs \mathbf{opus} with functions	126
5.1	Performance of \mathbf{opus}_β vs \mathbf{opus} with extensible records	200
5.2	Performance of \mathbf{opus}_β on object-oriented examples	201
C.1	The \mathbf{opus} algorithm	235